
ARTINFO MUSINFO

KRWTH

Jérôme Chailloux

KRWTH est un système conversationnel permettant d'exploiter un synthétiseur monodique construit au département d'informatique par Jean Claude MAZEAU et Hervé OULIÉ.
Ce synthétiseur est connecté à un ordinateur CAB502B

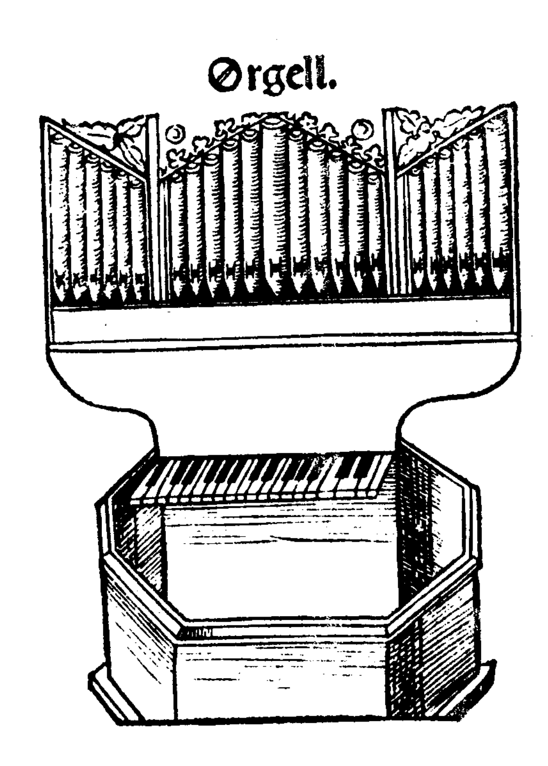
Description rapide de CAB502B

- ordinateur de seconde génération (1960)
- mémoire centrale de 32 Kmots de 32 bits sur tambour magnétique (temps d'accès au mot de 1/50 de seconde)
- 16 registres rapides
- entrées/sorties par :
  - machine à écrire émettrice-réceptrice (10 c/s)
  - perforateur de ruban (10 c/s)
  - lecteur de ruban (10 c/s et 200 c/s)
- programmation aisée :
  - instruction sur 32 bits
  - 13 opérations élémentaires
  - 32 extra-codes ("microprogrammés")
  - adressage registre-mémoire immédiat, direct ou indexé

Description (tout aussi rapide) du synthétiseur

Le synthétiseur est connecté en permanence à l'un des registres rapides (le registre 2). A chaque modification du registre 2, ses 32 bits sont transférés au synthétiseur et un son est émis (après décodage).
Pour chaque son, il faut mettre dans le registre 2 :
- une puissance (niveau)
- une hauteur (fréquence)
- un timbre

Remarques :
- le synthétiseur est monodique (il ne peut émettre qu'un son à la fois)
- c'est à l'ordinateur de prendre en charge les temps d'attente entre chaque transfert.



LA PROGRAMMATION KRWTH

L'utilisation du synthétiseur doit se faire en deux temps :
- chargement en mémoire d'une suite de sons, au moyen des <u>commandes</u>
- exécution de cette suite de sons

Le chargement effectif des suites de sons se fait dans 32 <u>sons programmes</u> (partition de la mémoire centrale), ce qui permet d'avoir jusqu'à 32 suites présentes en mémoire centrale, et offre la possibilité d'appel entre ces sons programmes.

Une commande est une suite de caractères (différents de l'espace) qui sont tapés sur la machine à écrire ou lus sur le lecteur de ruban, et interprétés immédiatement.

Pour mémoriser un son, il faut :
- le construire (puissance, hauteur, timbre)
- lui donner sa durée au moment de l'exécution
- le charger effectivement.

Ceci permet les "hésitations" au moment de la construction du son.

Il y a donc 5 types de commandes :
les commandes
- de construction de sons
- de durée de sons
- de chargement
- de gestion des sons programmes

et un ensemble de commandes générales.

I les commandes de construction de sons

Elles ont la forme suivante : un symbole (lettre mnémonique de la commande) suivie d'un attribut numérique qui peut être :
- un nombre en valeur absolue (a)
- un nombre relatif (±r) donnant à l'attribut la valeur du dernier attribut enregistré pour ce symbole ±r.

Toutes les commandes de ce type sont rémanentes, i.e. si certaines de ces commandes restent inchangées d'un son sur l'autre, il est inutile de les repréciser.

Après chaque commande le son modifié est entendu, ce qui permet des corrections avant changement du son construit.
On a vu qu'il fallait fournir au synthétiseur une puissance, une hauteur, un timbre.

Programmation de la puissance

Il y a 16 niveaux de sortie, linéaires

niveau 0 1 2 3 4 5 -------- 12 13 14 15

ppp pp p mf f ff fff

Symbole : P. attribut = niveau souhaité (entre 0 et 15)
ex : P11 signifie que le son en construction devra avoir le niveau 11.
P⊓ est interprété comme P0 (silence)
P+⊓ est interprété comme P+0 (puissance inchangée)
Une suite de commandes P4 --- P+8 --- P-6 --- P-6 --- P+14
sera équivalente à : P4 --- P12 --- P6 --- P0 --- P14

Programmation de la hauteur

Elle se fait au moyen de 2 commandes
- l'octave
- la note dans l'octave

• Octave

Il y a 8 octaves numérotées de 0 à 7

octave 0 1 2 3 4 5 6 7

grave → aigu

Symbole : H attribut = numéro de l'octave

• Note

Chaque octave peut être découpée en un certain nombre d'intervalles égaux naturels (voir la commande générale = Gp).

Symbole : N attribut = numéro de note

Si par exemple l'octave a été découpée en 12 sections (=G12) les numéros de notes seront les suivants :

	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11
équivalent à:	do	do#	ré	ré#	mi	fa	fa#	sol	sol#	la	la#	si

La suite de hauteurs

mi3 sol3 fa4 do4 do3 la2 mi4

pourrait se programmer : H3N4.... N7... H4N5... N0.... H3... H2N9... H4N4

ou bien : H3N4... N+3... N+10... N-5... N-12... N-3... N+19

Programmation du timbre

Ce n'est pas une véritable synthèse de timbres (par addition d'harmoniques) mais une possibilité de modification d'amplitude (niveau) et de fréquence à l'exécution du son. Ces modifications sont réalisées par des microprogrammes câblés, internes au synthétiseur, que l'on peut appeler.

• Les microprogrammes d'amplitude

Il y a 4 microprogrammes qui permettent une variation de niveau à l'attaque du son.

Symbole : A · attribut = numéro de microprogramme

P demandée ..... ←T→ ..... P0 .....

A0 amplitude croissante

PD ..... ←T→ ..... P0 .....

A1 amplitude croissante, puis stable

PD ..... ←T→ ..... P0 .....

A2 amplitude décroissante

PD ..... P0 .....

A3 amplitude constante

La durée (T) des différents micro-programmes n'est pas la durée du son mais la durée câblée propre au synthétiseur (voir commande suivante).

• Constante de temps du synthétiseur

Symbole : K attribut = n° d'une des 4 constantes de temps

K0	durée du microprogramme	= 1 x t	≈ 1/25 seconde
K1	durée du microprogramme	= 4 x t	≈ 1/6 seconde
K2	durée du microprogramme	= 16 x t	≈ 2/3 seconde
K3	durée du microprogramme	= 64 x t	≈ 3 secondes

• Les microprogrammes de fréquence

Ils permettent une modification de la fréquence (hauteur) pendant toute la durée du son, à vitesse fixe.

Symbole : M attribut = n° du microprogramme

M0 stabilité ⟶

M1 décroissance continue ↘

M2 croissance continue ↗

M3 alternance (tremolo) ∿∿∿...↗

II La commande de durée de son

Elle donne la durée du son au moment de l'exécution. Cette commande est également rémanente

Symbole : Δ suivi d'un des 3 attributs suivants :

a = nb de 1/50 seconde en valeur absolue
±r = nb de 1/50 seconde relatif à la dernière durée enregistrée
cmcm .....cm = une suite de symboles de durée programmée (C) suivie d'attribut de durée programmée (m)

Les symboles de durée programmée sont :
R B N C D T Q
équivalents à [notes de musique : ronde, blanche, noire, croche, double croche, triple croche, quadruple croche]

avec les attributs suivants :
[0 / 1] durée programmée normale
2 durée programmée pointée •
3 durée programmée de triolet ⌐3¬

Exemple de durées avec attributs numériques :

	Δ20	Δ+10	Δ-5	Δ40	Δ-20
nb. de 1/50 s.	20	30	25	40	20

Exemple de durées avec attributs programmés :
ΔN .... ΔBB2 ... ΔN3CT2
équivalant à : [notation musicale]

Les durées absolues des durées programmées sont calculées après initialisation (en 1/50 sec.) de la quadruple croche (voir la commande générale =Q9).

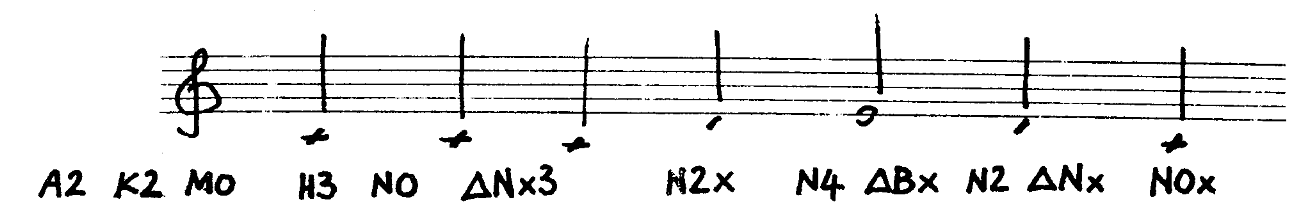
III La commande de chargement

Commande de chargement de son

Après avoir construit le son et donné une durée, il faut mémoriser ce son. Cette mémorisation à lieu dans un des sous programmes, le s.p. courant de chargement (voir les commandes suivantes).

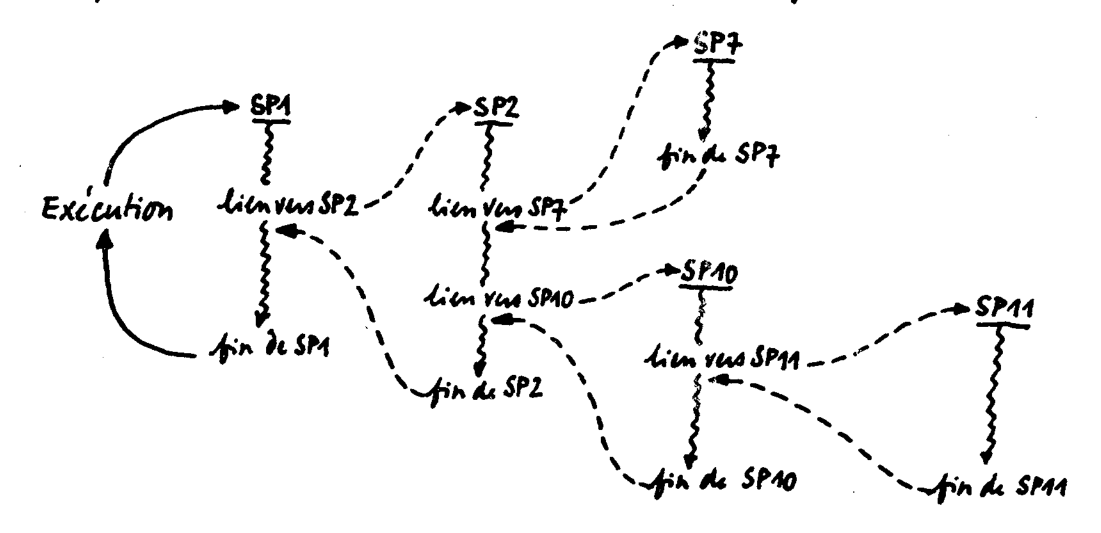
Symbole : × (le caractère de multiplication, pas la lettre X)
Attribut = nombre de fois qu'il faut charger la note
si cet attribut est égal à 1, on peut l'omettre

Exemple (sans tenir compte des s.p.):



Commande de "lien vers sous programme"

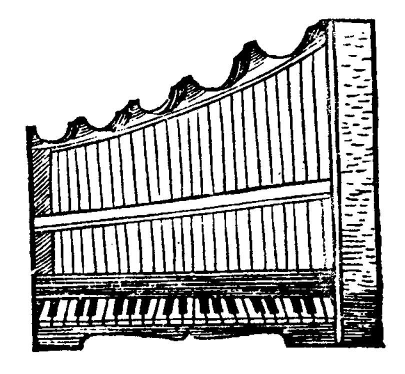
On peut également charger des "liens vers s.p." qui, au moment de l'exécution, aiguillent vers un autre s.p.. A la fin du s.p. appelé, on retourne au s.p. appelant.



Programmation : x k L i
k est le nombre de fois qu'il faut charger un lien vers le sous-programme i.
Si k = 1, on peut l'omettre.

Les adresses de retour étant stockées dans une pile de neuf registres, on ne peut dépasser 9 niveaux d'incrustation, ni faire de s.p. récursifs (contenant des liens avec eux-mêmes). En cas de dépassement de capacité de la pile au moment de l'exécution, un déroutement se produit vers un s.p. spécial qui exécutera indéfiniment le Dies Irae, signalant une erreur d'exécution.

Clauicyterium.



IV Commandes de gestion des s.p.

Il y a 32 sous-programmes numérotés de 0 à 31

Les commandes sont :

/Di début du s.p. i qui devient le s.p. courant
/Ci continuer le s.p. i
/F finir (fermer) le s.p. courant de chargement
/Ei entendre le s.p. i

Si i n'est pas compris entre 0 et 31, le message suivant apparaît : ERREUR NO SP (numéro erroné).

Si des commandes de chargement apparaissent avant la définition du sous-programme courant ou après la fermeture de celui-ci, le s.p. 0 est ouvert automatiquement (/DO implicite).

Si les commandes /Di, /Ci ou /Ei apparaissent avant la fermeture du s.p. courant antérieur, celui-ci est fermé automatiquement, et le message suivant est imprimé : FIN DU SP (n° du s.p. fermé).

V Commandes générales

Découpage de l'octave

Commande : =Gp
p est le nombre d'intervalles que comporte chaque octave (cf. la programmation des hauteurs. p doit être compris entre 1 et 128.
Temps d'exécution : 4 secondes

Exemples :
=G12 définit une échelle chromatique
=G5 définit une échelle pentatonique

Initialisation des durées programmées

Commande : =Φq
q est la durée en cinquantièmes de seconde de la quadruple croche normale (cf. la programmation des durées).
Temps d'exécution : deux secondes.

Exemple :
=Φ2 signifie que la 𝅘𝅥𝅱 normale dure 2/50 seconde, et

implique : 𝅘𝅥𝅱 𝅘𝅥𝅰 𝅘𝅥𝅯 𝅘𝅥𝅮 𝅘𝅥 𝅗𝅥 𝅝

2/50 4/50 8/50 16/50 32/50 64/50 128/50

Si 𝅘𝅥 = 32/50 alors 𝅘𝅥 = 94 (94 à la noire)

Introduction de constantes standard

Commande : =K
Charge automatiquement la suite de commandes :
=G12 =φ2 P12 H3 NO A2 K2 MO ΔN
Temps d'exécution : 1 seconde

Halte

Commande : =Hl
est équivalente à l'instruction FORTRAN PAUSE (N)
avec impression des valeurs actuelles des commandes de construction de sons, et des commandes de durée.

Restauration générale du système

Commande =Z
Remet à zéro tous les sous programmes et réinitialise le système.
Temps d'exécution : 4 à 5 secondes.
Cette commande doit être utilisée après accord des autres utilisateurs, car tous les s.p. sont perdus.

Virginal.

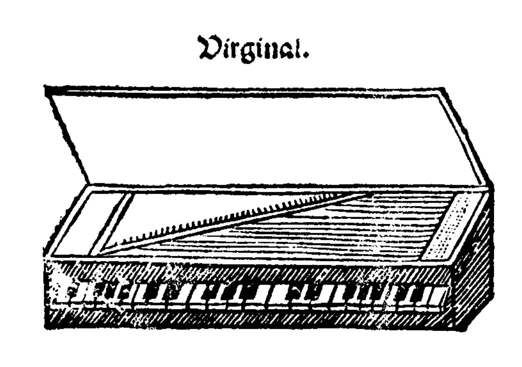


Tableau récapitulatif des commandes KRWTH

Construction de sons

Commande	Domaine	Signification	Commande	Domaine	Signification
$P\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq 15$	Puissance	$A\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq 3$	microprogramme amplitude
$H\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq 7$	Octave	$K\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq 3$	constante de temps microprog.
$N\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq p-1$	note (avec p défini par =Gp)	$M\begin{Bmatrix} a \\ \pm z \end{Bmatrix}$	$0 \leq a \leq 3$	microprogramme fréquence

Durée de sons

$$\Delta\begin{Bmatrix} a \\ \pm z \\ c m\, c m \ldots\ldots c m \end{Bmatrix} \qquad 0 \leq a \leq 131071 \qquad c = \begin{Bmatrix} R \\ B \\ N \\ C \\ D \\ T \\ Q \end{Bmatrix} \qquad m = \begin{Bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{Bmatrix}$$

Chargement

Commande	Domaine	Signification
Xk	$0 \leq k \leq 131071$	charger k fois la note construite
XkLi	$0 \leq i \leq 31$	charger k fois un lien vers le s.p. i

gestion des s.p.

Commande	Domaine	Signification	Commande	Domaine	Signification
/Di	$0 \leq i \leq 31$	début s.p. i	/F		finir s.p. courant
/Ci	$0 \leq i \leq 31$	continuer s.p. i	/Ei	$0 \leq i \leq 31$	entendre s.p. i

Commandes générales

Commande	Domaine	Signification
=Gp	$0 \leq p \leq 127$	découpage octave
=Qq	$0 \leq q \leq 131071$	initialisation quadruple croche
=K		introduction de cotes. standard
=Hl		halte + impression
=Z		restauration générale système

MISE EN ŒUVRE DU SYSTÈME [illegible]

① Enclencher l'interrupteur général ([illegible] salle à gauche de l'ordinateur). Bouton vert enfoncé

② Appuyer sur le voyant rouge pupitre "DÉMARRAGE" qui doit s'allumer. Attendre quelques minutes que l'ordinateur soit synchronisé.

③ Vérifier que les pistes 0 à 175 sont déverrouillées (boutons en position haute, sous le carter à droite)

④ Allumer la flexo :
- inverseur "ARRET-MARCHE" sur "MARCHE"
- inverseur "I - I+P - P" sur "I"
- boutons-poussoirs "TRANS" et "RECEPT" en position basse

⑤ Vérifier que le voyant-pupitre "KRWTH-UNI" est en position "KRWTH", sinon appuyer une fois sur ce poussoir

⑥ Appuyer sur le bouton-poussoir "DEPART"
Le système écrit sur la flexo "Σ⊓ :". Frapper la lettre K.
Le système répond RWTH, fait un retour chariot et écrit "?"
Le système est prêt à recevoir des commandes.

- Pour rappeler le système en cours d'exécution (audition) ou en cas d'erreur grave, reprendre en ⑥

- En cas d'erreur diagnostiquée, l'ordinateur s'arrête en "fin". Appuyer sur "MARCHE" pour continuer l'interprétation.

- En règle générale, frapper doucement les commandes à la flexo et faire un espace entre chaque commande

- Pour tous renseignements ou suggestions, s'adresser à Jérôme CHAILLOUX, Dept. d'Informatique, Université Paris VIII, Route de la TOURELLE, 75571 PARIS CEDEX 12.

EXEMPLES

Σ: KRWTH

```
? =K
/D1 A1 K1 ΔC H1 N7×
N+7× N+9× N-2× N+2× N-9× N+9× N-9× /F
              FIN DU SP  1

/D2 ΔC H1 N7×
N+9× N+8× N-1× N+1× N-8× N+8× N-8× /F
              FIN DU SP  2

/D3 ΔC H1 N7×
N+11× N+6× N-1× N+1× N-6× N+6× N-6× /F
              FIN DU SP  3

/D4 ×2L1   ×2L2   ×2L3   /E4
              FIN DU SP  4
```

Σ: KRWTH

```
? /C4 ΔC H1 N7× N+12× N+4× N-2× N+2×
N-4× N+4× N-4× H1 N7× N+12× N+4× N-2×
N+2× N-4× N+4× N-5×
P() ΔN ×  /E4
              FIN DU SP  4
```

Σ: KRWTH

```
? /D5  P15 A() K3 H2 N11 ΔB ×
N9 ΔN × N11 × N4 ΔR × N11 ΔN ×
N9 × N7 × N6 × N7 × N9 × N7 ×
N6 ΔN2 × N4 ΔR × N2 ΔN × N4 × N6 ×
N7 × N9 × N4 × N6 × N2 × N4 ΔB2 ×  /E5
              FIN DU SP  5
```

Σ: KRWTH

```
? /D6  ×2L4  ×1L5  /E6
              FIN DU SP  6
```

COMPUTER ART : UNE PARTITION POUR LE PEINTRE

P.L. NEUMANN

INTRODUCTION

De l'idée à "l'expression" d'une oeuvre par ordinateur il y a un programme. Si ce programme n'est qu'un exécutant la création et donc "l'expression" de l'artiste ne s'exerceront qu'au niveau du programme. Pour qu'il y ait "expression" dans le résultat il faudrait que le programme soit l'artiste ou que l'artiste travaille avec son programme.

C'est dans cette dernière perspective que se situe le programme "INTERPRETATION".

Ce programme ne réalise rien, mais propose un programme pour l'artiste : une partition à interpréter.

Ce programme a été écrit en CONNIVER - LISP en Mars 1974.

Comment fonctionne ce programme ?
On lui donne une idée de base ou contexte de la forme :

(A sur B)
(C devant A)

c'est à dire un couple d'objets non définis, liés par une relation x quelconque telle qu'il existe son contraire $\bar{x}$.
A partir de cette ébauche le programme va développer une partition c'est à dire un certain nombre de propositions telles que (A x B) - A, B et x éléments du contexte.
Chaque étape de cette partition correspondra donc au développement d'un objet du contexte par rapport à un autre objet de ce même contexte, le choix d'un tel ou tel objet et d'une relation ne dépendant que de l'état courant du contexte à un instant donné.
En effet, chaque proposition du programme devra être cohérente par rapport à l'idée de base : l'union de la proposition (p) et du contexte (c) devra être égale au contexte (c) lui - même, puis (p) sera ajouté à (c), produisant ainsi un nouveau contexte permettant le développement de nouvelles relations.
Le processus se déroulera ainsi jusqu'à ce que l'idée de base n'offre plus aucune possibilité de développement.

Le second travail va maintenant consister à interpréter la partition ainsi construite : interpréter voulant dire dans le cas présent matérialiser les objets et les relations de la partition en vue d'une exécution en pas à pas, respectant l'ordre chronologique de cette partition.
Autrement dit, il s'agit en fait de concrétiser les éléments d'un algoritme produit par un programme et dont on aura auparavant précisé l'idée de base, puis l'exécuter à la main ou par programme.

LE PROGRAMME

I - Développement de l'idée de base ou contexte.

a) L'idée de base

Prenons comme exemple la liste des propositions suivantes correspondant pour le peintre à une esquisse de son dessin :

(A à gauche de B)
(A à gauche de C)
(B sur C)

et la liste de toutes les relations employées et de leurs inverses:

```
( Setq  SIT  ' ( à gauche de  à droite de  sur  sous ))
```

.../...

Le programme va commencer par analyser ce contexte en y ajoutant les informations suivantes :

1 - <u>Hypothèse de Travail</u>:
" ne sont définis par relation que les objets A et B , en effet dans une relation du type (α x β) j'ai supposé que l'objet B était l'objet " subissant " la relation x et que donc, seul l'objet α était défini."

2 - <u>Etat des Objets mis en relation</u>:
(/ à droite de A) il doit y avoir quelquechose
(/ sous B) " " " " "
C n'est pas défini.

3 - <u>Idée ou Pattern à respecter</u>:
le graphe

A ⟶ B, A ⟶ C, B ⟶ C

ou la liste

((A B) (A C) (B C))

4 - <u>Une liste</u> comportant tous les objets et leur relation qui permettra de déterminer les objets à développer. Cette liste ne fera pas partie du contexte : elle représentera l'état de chaque objet.

(A est à gauche) de quelquechose
(B est à droite) " "
(B est sur) " "
(C est sur) " "
(C est à droite) " "

Après analyse le contexte initial sera représenté par la liste:

((A à gauche de B)
(A à gauche de C)
(B sur C)
(/ à droite de A) (/ sous B)
(A B) (A C) (B C))

b) <u>Développement du contexte</u>

Cette tâche va consister à produire une ou plusieurs propositions du type (α x β) respectant le pattern du contexte. Pour cela, le programme va se poser un certain nombre de questions.

1 - <u>Quel objet développer et vers quel autre ?</u>

Pour cela on ira consulter la liste de l'etat des objets (mentionnée en 4- du a)). Les objets a développer seront les objets qui auront le même état. En suivant notre exemple, ces objets seront B et C puisqu'ils possédent le même état: <u>est à droite de</u> .

.../...

Par ailleurs on s'assurera que la proposition (C B) n'appartient pas déjà au pattern du contexte. Dans le cas contraire on inverse cette proposition qui deviendra (B C). En fait, je pense à un peintre qui aurait à 'travailler des masses égales' pour les intégrer au reste de son dessin. Si aucun objet n'a le même état ou que toutes les propositions appartiennent déjà au pattern, on prendra alors des objets de la dernière proposition du contexte : le peintre continue à développer son dessin.
- Toute cette première partie n'a pas été programmée par manque de place en mémoire, mais elle n'offre aucune difficulté -

2 - <u>Quelle relation appliquer à ces deux objets.</u>

Cette relation sera déterminée par le pattern du contexte : Un peintre construit sa toile en appliquant certaines touches qui doivent s'harmoniser avec le reste de son dessin. Le programme en fait autant: En effet il devra trouver la relation x entre deux objets (C et B dans notre exemple) telle que :

- A partir de Cx on retrouve ce qui est lié à C dans le pattern.
- A partir de $B\bar{x}$ ($\bar{x}$ inverse de x) on retrouve ce qui est lié à B dans le pattern.

C'est à dire que si un objet α est lié à l'élément du pattern (α γ) alors, à la proposition α x devra correspondre un (/ x α) dans le contexte.

Ex : Cx ⟶ rien (et strictement rien)
$B\bar{x}$ ⟶ (B C)

Ce qui est lié aux deux objets OX et OY dans le pattern sera fourni par la fonction TRAJET .

```
(DE TRAJET  ( OX OY ;; OZ OW )
        (FOR-EACH ( !OZ !OW )
          (COND
               ((EQ OZ OX)
                (SETQ R-OX
                     (CONS(LIST OZ OW) R-OX)))
               ((EQ OZ OY)
                (SETQ R-OY
                     (CONS(LIST OZ OW) R-OY)))
          )))
```

.../...

Puis on essayera toutes les relations possibles jusqu'à ce qu'il y en ait une qui convienne.

```
(DE ESSAI    ( OX OY ;; P PREDIC )
             (SETQ AJOUT (SETQ R-OX (SETQ R-OY NIL)))
             (TRAJET OX OY)
             (PRINT =( TRAJET ,R-OX ,R-OY ))
             (SETQ PREDIC SIT)
             (WHILE PREDIC
                    (SETQ P (NEXTL PREDIC))
                    (AND (OR (NONSIT OX P R-OX)
                             (IS OX P R-OX))
                         AJOUT
                         (ASSUMING @AJOUT
                                   (OR (NONSIT OY (NON P) R-OY)
                                       (IS OY (NON P) R-OY)))
                         (NULL AJOUT)
                         (PRINT (ADD '( DO @OX @P @OY )))
                         ))
              (PRINT '//////))
```

Pour chaque essai le programme supposera que la proposition (C x B) est la bonne et pour cette raison il créera un contexte hypothétique (ASSUMING ......) composé du contexte courant <u>plus</u> l'état de l'objet défini par supposition c'est à dire (/ $\bar{x}$ C) : le peintre ajoute un élément à son dessin <u>puis</u> regarde s'il n'en détruit pas l'harmonie.

```
(DE REPONS  ( OX P )
   (SETQ AJOUT
         (COND
             (AJOUT NIL)
             ((LIST '/ (NON P) OX)) ) ))
```

ramène la liste (/ $\bar{x}$ OX).

.../...

```
(DE NONSIT  ( OX P ROX ;; OZ )
            (OR ROX
                (PRESENT '( / @P !OZ ))
                (REPONS OX P)))
```

Si l'objet OX n'est lié à aucun élément du pattern
c'est à dire Cx ⟶ rien .

```
(DE IS      ( OX R ROX ;; OZ HYP )
            (COND
                (ROX (FOR-EACH ( / @R !OZ )
                                (SETQ HYP (CONS =( @OX ,OZ ) HYP)))
                    (AND (RESPEC ROX HYP)
                         (REPONS OX R)))))
```

Essai d'une relation.

```
(DE RESPEC  ( RO HYP )
    (COND
        ((NULL RO) 'OK)
        ((MEMBER (NEXTL RO) HYP)
         (RESPEC RO HYP))))
```

Vérifie si la relation proposée respecte le pattern.
Dans notre exemple on trouvera :
(C sur B) en effet

- contexte hypothétique = contexte courant + (/ sous C)
- C sur ⟶ rien
- B sous ⟶ (B C)

.../...

puis paralellement (plusieurs relations peuvent satisfaire la même proposition)

(C à gauche de B)

Le contexte courant sera alors modifié en ajoutant définitevement :

- les propositions (C sur B)
  (C à gauche de B)
- l'état des objets mis en relation s'il ne s'y trouve pas déjà :
  (/ sous C)
  (/ à droite de C)
- le nouvel élément du pattern :
  (C B)

Tout ceci sera le travail de la fonction REMPLACE .

```
(DE REMPLACE  ( ;; OX R OY )
   (COND
       ((PRESENT '( DO !OX !X !OY ))
        (FOR-EACH ( DO !OX !R !OY )
                  (REMOVE '( TRY ,OX VERS ,OY ))
                  (REMOVE '( DO ,OX ,R ,OY ))
                  (AND (ABSENT '( / @(NON R) ,OX ))
                       (ADD '( / @(NON R) ,OX )))
                  (AND (ABSENT '( ,OX ,OY ))
                       (ADD '( ,OX ,OY ))))
       'OK )))
```

La liste de l'état des objets sera également modifiée en y ajoutant de nouveaux états :

Ex: (C est sur)
(C est à gauche)
(B est sous)
(B est à gauche)

A partir de ce nouveau contexte on pourra développer de nouvelles propositions.

Dans le cas où aucune relation n'aurait été trouvée le programme précisera les éléments du pattern liés aux objets de la proposition, par exemple :

Si aucune relation n'est trouvée pour la proposition (α $\times$ β) celle ci sera alors insérée au contexte courant sous la forme (TRY α vers β) puis sachant que α est lié à (α γ) et que β est lié à (β γ_1) et (β γ_2) on développera chaque élément de la liste :

EXPLOR = ((α γ) (β γ_1) (β γ_2))

.../...

```
(DE SEARCH  ( EX )
   (PROG ()
      B  (OR EXPLOR (RETURN))
         (SETQ EX (NEXTL EXPLOR))
         (COND
             ((ABSENT '( TRY @(CAR EX) VERS @(CADR EX) ))
              (PRINT (LIST 'EXPLORE (CAR EX) (CADR EX)))
              (ESSAI (CAR EX) (CADR EX)))
             ((GO B)))
         (AND (REMPLACE)
              (END)
              (RETURN 'OK))
         (GO B)))
```

Ainsi $(\alpha\ x\ \gamma)$ devient une nouvelle proposition et le programme devra trouver la relation x telle que

$$\alpha\ x \longrightarrow \text{(élément du pattern)}$$
$$\gamma\ \bar{x} \longrightarrow \text{(élément du pattern)}$$

Si une relation est trouvée, alors le contexte sera modifié, puis on essayera à nouveau la proposition $(\alpha\ x\ \beta)$

```
(DE END      ( OX OY )
   (PROG ()
      A  (COND
            ((ABSENT '( TRY !OX VERS !OY ))
             (RETURN 'OK))
            (T (FOR-EACH ( TRY !OX VERS !OY )
                         (PRINT CURRENT)
                         (ESSAI OX OY))
               (OR (REMPLACE) (RETURN))
               (PRINT '------)
               (GO A)))))
```

Cette fonction ira chercher toutes les propositions du type (TRY α vers β) et essayera de les développer.

.../...

En conclusion, le programme ne s'arrêtera que lorsque tous les objets auront les mêmes états ou lorsqu'il n'y aura plus aucune possibilité de développement. On obtiendra alors une partition .

D'autre part, en raison de l'absence de la fonction (décrite en 1 - du b) page 2) permettant le lancement du programme, on devra préciser pour cela le ou les couples d'objets à développer en les insérant au contexte initial sous la forme (TRY α vers β).

```
(DE DEVELOPPE  ( OX OY ;; LPOS )
   (COND
       ((END) 'NEW)
       (T (SETQ LPOS (FETCH '( TRY !OX VERS !OY )))
          (ESCAPE EXIT
                  (WHILE LPOS
                         (TRY-NEXT LPOS)
                         (PRINT =( EXPLORER ,OX VERS ,OY ))
                         (SETQ R-OX (SETQ R-OY NIL))
                         (TRAJET OX OY)
                         (SETQ EXPLOR (NCONC R-OX R-OY))
                         (AND (SEARCH (EXIT 'NEW)))))))
```

.../...

c) La partition

Cette partition sera composée d'un certain nombre de propositions réparties suivant un ordre bien précis correspondant à l'ordre des possibilités de développement. A partir de l'idée de base donnée en exemple on obtiendra la partition suivante :

(A GAUCHE B)
(A GAUCHE C) idée de base
(B SUR C)

(C SUR B)
(C GAUCHE B)
(C SOUS B)
(A SOUS B)
(B SUR A)
(B SOUS A)

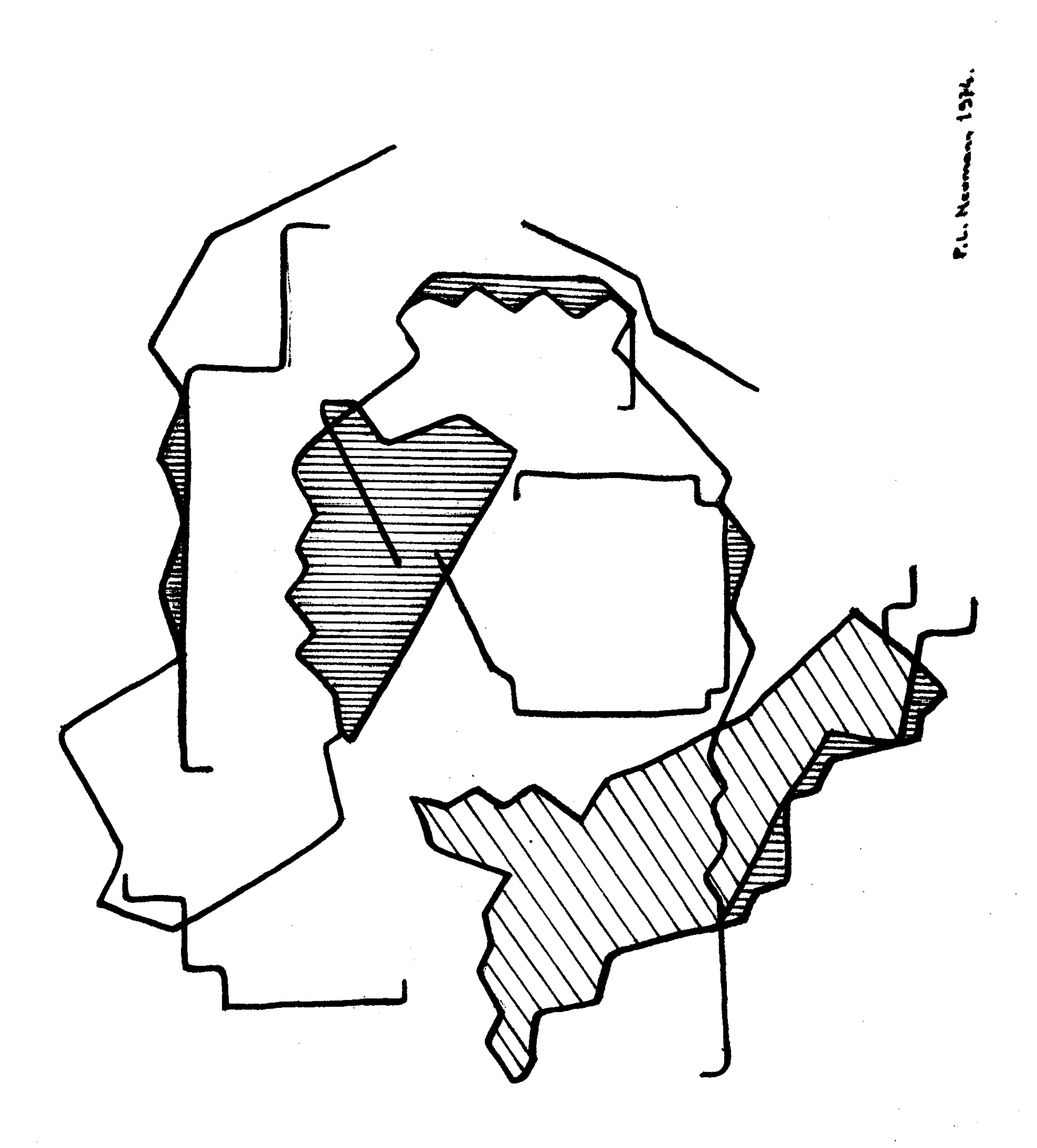
(B SOUS A)
(A GAUCHE C) idée de base
(C SUR B)

(B DROITE C)
(C SUR A)
(C GAUCHE A)
(A SOUS C)
(A DROITE C)

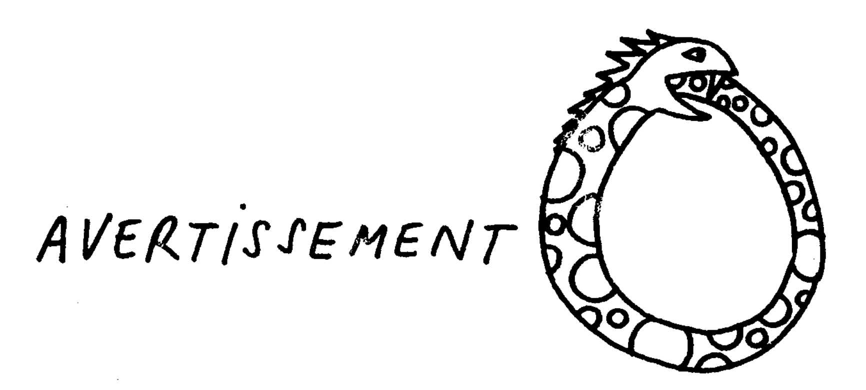
(C GAUCHE A)
(B DROITE C) idée de base
(A SOUS B)

(A GAUCHE B)
(B DROITE A)
(A GAUCHE C)
(C DROITE A)
(B DROITE A)

qu'il faudra INTERPRETER ....


P.L. Newman 1974.

P.L. Neumann 1974



Le présent bulletin répond à une visée toute didactique : livrer sous forme accessible aux nouveaux venus dans les groupes de travail courants

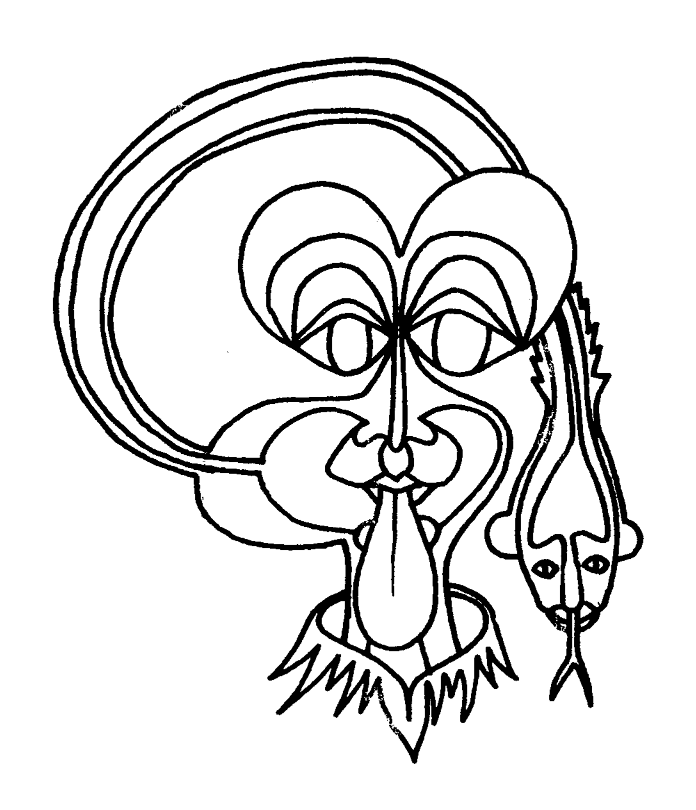
- de l'information technique et bibliographique en rapport avec leurs disciplines
- des programmes commentés de tous niveaux permettant un accès relativement rapide à des techniques de programmation appropriées, ainsi qu'à une implémentation aisée.

On s'est efforcé, dans la mesure du possible, de ne pas établir de clivage trop net entre les disciplines concernées (musique, arts plastiques, poésie, architecture, logique, informatique), mais tout au contraire de les unifier, ne serait-ce que par des techniques de programmation communes.

L'aspect pédagogique d'ARTINFO/MUSINFO reflète une préoccupation constante du groupe, à savoir ne pas se satisfaire en dernier ressort de méthodes de programmation trop élémentaires.

Pour tous renseignements et composition des livraisons à venir, s'adresser à Jacques ARVEILLER, Département d'Informatique, Université de PARIS VIII, Route de la Tourelle, 75571 **PARIS CEDEX 12**
Pour tout envoi, s'adresser à Patrick GREUSSAY, même adresse.

ARTINFO/MUSINFO est imprimé au Département d'Informatique de l'Université de PARIS VIII.



ARTINFO / MUSINFO #20
Université Paris VIII
Institut d'Intelligence
Artificielle
groupe Art & Informatique